Appendix A

BIOS Power Management Service for Windows Nt 4.0

Introduction

This document provides the interface for the 32-Bit BIOS Power Management Service Interface (BPMSI). This interface is used by the Windows NT Power Management Kernel driver for providing power management services to APM-aware applications.

Terms

APM BIOS

System BIOS which provides power management functions adhering to the APM specification (currently at revision level 1.2)

Kernel Mode

A privileged processor mode in which the NT system code runs. A kernel mode thread has access to all I/O and system memory

Power Management Kernel Driver (PM Driver)

Provides access to the BIOS ROM, BIOS Data Area and the 32-Bit Services for Services.

Power Management Service (PM Service)

Provides power management services for applications and other drivers. Translates their requests into queries to the BIOS Power Management Service Interface using the interface provided by the Power Management Kernel Driver (PM Driver).

PowerPAL

Phoenix's APM BIOS extensions which provide application level access to system and device power management features

System Idle

State where PM service detected minimum processing at the application level. In this state, only threads run on idle priority are executed and they will be preempted by any thread running in a higher priority class.

User Mode

A non-privileged processor mode in which application code runs. An user-mode thread does not have access to I/O and system memory.

Architecture Overview

The extensions to the BIOS for support of Windows NT power management consist of two parts:

(a) Modifications to the BIOS32 Service Directory. An entry was added which allows the PM Driver to find the entry point for the BIOS Power Management Service Interface.
(b) New 32-Bit BIOS Power Management Service, which mimics the APM 32-bit interface with some differences, which are noted below. The new interface provides a 0:32 calling method which is more secure and more Windows NT friendly.

When the PM Kernel wishes to use the BIOS services, it must perform the following steps:

1. Find the BIOS32 Service Directory header
2. Call the BIOS32 Service Directory calling interface, specifying the 32-Bit BIOS Power Management Service Interface entry point.
3. Call the BIOS 32-Bit BIOS Power Management Service Interface entry point, asking for APM Connect.

BIOS32 Service Directory Modifications

The 32-bit BIOS Service Directory is an existing structure within the Phoenix BIOS which allows a 32-bit protected mode application or operating system to find the entry point for a particular 32-bit service. This specification defines a new standard 32-bit BIOS Service.

The BIOS32 Service Directory consists of a fixed structure which can be detected by the PM driver and a single function which returns the address for a particular service.

The BIOS32 Service Directory Header

A BIOS which implements the BIOS32 Service Directory must embed a specific, contiguous 16-byte pattern somewhere in the physical address range 0E0000h - 0FFFFFh. The pattern must be paragraph aligned (i.e., it must start on a 16-byte boundary). This pattern is known as the BIOS32 Service Directory Header.

The Header is comprised of six distinct fields. The following table describes each field.

Offset	Size	Description
0	4 bytes	The ASCII signature "_32_" or 0x5F33325F
4	4 bytes	The entry point for the BIOS32 Service Directory calling interface. This is a 32-bit linear (i.e., not segment:offset) physical address
8	1 byte	The revision level of the BIOS32 Service Directory header and calling interface. The current revision is 0.
9	1 byte	The length of the BIOS32 Service Directory header. Measured in paragraphs (16 bytes). The current length is 1.
10	1 byte	The BIOS32 Service Directory header byte-wide, byte-sum checksum. When totaled, all of the bytes in the header should ad up to 0.
11	5 bytes	Reserved. Should be 0.

Clients of the BIOS32 Service Directory should first determine its existence by locating the Header. This is done by scanning 0E0000h to 0FFFF0h in paragraph increments and looking for a signature match ("_32_") in the first 4 bytes of each paragraph. When, and if, the signature is detected the client should perform a checksum of all bytes in the Header. (The Header length, in paragraphs, is found at offset 9h.) All bytes in the Header should ADD together with a result of 0h. If the checksum is valid then the 32-bit entry point

field can be used as the address for the BIOS32 Service Directory Calling Interface. If the Header is not found then the BIOS32 Service Directory does not exist on the platform.

The BIOS32 Service Directory entry point, and its associated code and data, maybe located anywhere within the 4Gb physical address space. However, it is guaranteed to be physically contiguous (i.e., it will be delivered in ROM or FLASH space) and to fit within two pages (i.e., it will not span three pages).

The BIOS Service Directory

To get the entry point to the BIOS 32-Bit Power Management Service Interface, the PM Driver must call the BIOS Service Directory with the following parameters:

IN:	EAX	"NTPM" or 0x4E54504D
	EBX	0x00000000
OUT:	AL	Error code: 0x00 = None, 0x81 = Service does not exist
	EBX	Base address of the 32-Bit Power Management Service code
	ECX	Length of the 32-bit Power Management Service code (from EBX)
	EDX	Offset (from EBX) of the 32-bit Power Management Service code entry point.

The entry point of the 32-bit Power Management Service code entry point is FAR (that is, requiring both segment and offset to be pushed on the stack.)

CS: The base address must be less than or equal to the (4KB) page address of the page that contains the entry point. For example, if the entry point is 0FFF81234h, then the base address must be less than or equal to 0FFF81000h. The limit must be such that the base address plus the limit generate an address that is greater than or equal to the last address on the (4KB) page which follows the page containing the entry point. For example, if the entry point is 0FFF81234h then the base address plus the limit must be greater than or equal to 0FFF82FFFh. Simply stated, the base address and the limit must "encompass" both the page that contains the entry point and the following page.

The segment type must be 100b (code, execute only) or 101b (code, execute/read). However, the implementers of the Service Directory cannot assume read access to the CS code segment. The system bit must be 1 (non-system segment). It is recommended that the Descriptor Privilege Level (DPL) be 0. (The CS descriptor DPL becomes the Current Privilege Level, or CPL). If the CPL is not 0, then the OS must provide trapping and virtualization services for ring 0 privileged instructions (such as those that access CRx). Note also the dependency of this field on the IOPL field in EFLAGS (see Section0).

The Default Size bit must be 1 (32 bits).

DS: The base address must be equal to the CS base address. The limit must be greater than or equal to the CS limit.

The segment type must be 000b (data, read only) or 001 (data, read/write). However, the implementers of the Service Directory cannot assume write access to the DS data segment. The system bit must be 1 (non-system segment). The Descriptor Privilege Level (DPL) must be greater than or equal to CPL (see the DPL field in Section 0).

SS: The segment type must be 011b (data, read/write, expand-down) or 001b data, read/write, expand-up). The system bit must be 1 (non-system segment). The Descriptor Privilege Level (DPL) must be equal to the CPL (see the DPL field in Section 0). The Default Size bit must be 1 (32 bits). The Granularity bit must be 1 (4Kb).

Note that the above settings ensure a stack size of at least 4kb. It is the caller's responsibility to ensure that there is at least 1kb of unused stack available.

Paging: Paging may or may not be enabled. If paging is enabled, then the address space that is described by the CS and DS selectors must be linearly contiguous. That is, the original physical contiguity of the Calling Interface as found in ROM or FLASH must be preserved. (The Calling Interface code and data is written to be position-independent and EIP-relative).

IOPL: In order for the Calling Interface to execute I/O instructions, the I/O Privilege Level (IOPL) field in EFLAGS must be greater than or equal to the CPL (see the DPL field in Section 0).

Other: The BIOS Data Area, Extended BIOS Data Area and fixed-location ROM data tables cannot be assumed to be available for use by the executing code because of paging. The BDA may be accessed using the pointer provided by the caller.

32-Bit BIOS Power Management Service Interface

In general, the 32-Bit BIOS Power Management Interface provides the same functionality as the 32-bit APM entry point. There are two areas of difference: the way in which calling parameters are passed and the way in which certain APM connect functions are supported.

Calling Parameters

The APM 32-Bit Interface passes all parameters in CPU registers. The BIOS Power Management Service Interface passes the parameters on the stack. The equivalent C-style declaration would be:

```
typedef struct
{
    ULONG           reserved0;      /* 00 */
    ULONG           pBDA;           /* 04 */
    ULONG           regFlags;       /* 08 */
    ULONG           reserved1;      /* 0C */
    ULONG           reserved2;      /* 10 */
    ULONG           reserved3;      /* 14 */
    ULONG           reserved4;      /* 18 */
    ULONG           reserved5;      /* 1C */
    ULONG           reserved6;      /* 20 */
    ULONG           reserved7;      /* 24 */
    ULONG           reserved8;      /* 28 */
    ULONG           reserved9;      /* 2C */
    ULONG           regEBP;         /* 30 */
    ULONG           regEDI;         /* 34 */
    ULONG           regESI;         /* 38 */
    ULONG           regEDX;         /* 3C */
    ULONG           regECX;         /* 40 */
    ULONG           regEBX;         /* 44 */
    ULONG           regEAX;         /* 48 */
} regStruct;

unsigned char BPMSI(regStruct* parameters);
```

The actual APM function and its behavior will be determined by the value written in the corresponding register fields. The pBDA field is a pointer to the virtual address where the kernel driver has mapped the BIOS Data Area so that it can be accessed by the service entrance.

If an error occurs, bit 0 of regFlags will be 1 and the error code will be in bits 8-15 of regEAX, otherwise it will be 0 and bits 8-15 of regEAX will be 0. The error codes are identical to those found in the APM 1.2 specification.

Software Requirements

Development Software Requirements

The following software development tools are required to build the PM service:

- MASM 6.11c assembler from Microsoft
- Phoenix PhDebug for debugging the code

Software Requirements

PM service requires to run on the following environment:

- Windows NT 4.0
- System with Phoenix NoteBIOS supporting the NT BIOS Interface to the PowerPAL
- PM driver installed in the system

Porting Changes

The following section describes the files which have been modified in the Core, Miser and the chipset code. It also describes the changes which need to be made in order to port a normal BIOS to include the NT support.